

Simplified Software Project Management for the Rest of Us Or a Twelve Step Program for the Chronically Overworked Programmer, Project Leader or Manager

Tom Mannigel , Insyst,Inc.

Abstract

Many SAS systems such as Data Warehouses are build by one programmer or a single team who are looking for all the help they can get. Unfortunately the so called “Rapid Development Techniques” designed to help quicken software projects are for the most part for huge teams spending millions of dollars on software that’s sold by the millions. For that reason these approaches are very complex and require a lot of resources. Something-smaller projects don’t have. There’s a big difference in the support and resources for a project that the president has bet the company’s success on and a “whatever’s needed” custom SAS system for a small group of very impatient end users. It’s like trying to build a custom home using the same approach you’d use to build a skyscraper. Unhappily you’re probable going to waste and not save time. There must be a simpler way. There is. “Simplified Software Project Management For the Rest of US”, describes a 12 step approach I’ve developed over the last couple of decades to help speed software projects for the rest of us who don’t have the luxury of infinite resources and time.

Introduction

The goal of this approach is to save time and increase your value. There may appear to be addition unneeded steps. But let me assure you every step is important and design to save time in the long run.

The twelve steps are with approximately percentage of the total project:

1. Do a preliminary design and estimate 2.5%
2. What’s it’s worth? 3%
3. What are the risks? 2%
4. Do a very detailed design and schedule. 15%
5. Build the system. 45%
6. Have weekly progress reports. 5%
7. Do systematic testing. 5%
8. Do transaction level testing. 10%
9. Do user testing. 5%
10. Document it quickly. 5%
11. Have a Party. 0%
12. Was it worth it? 2.5%

Now let’s go over each step in detail:

Step 1. Do a preliminary design and estimate.

What you do.

This is a step that must be done for all projects now matter how large or small. For this step create a quick generalize view of what the system would look like at a high level. Something like: a reporting system that reads x number of files and creates y number of reports. Based on the size and number of reports you make a guess as to how long it should take to do the project. Remember that this is a quick and dirty estimate. A more detail estimate will come later.

How long should you spend.

This step should take from five minutes to a few days based on the size of the project. For this and all the following examples, a small project is 1 programmer for six weeks and projects at the high end is a team of seven for 6 months.

How this step saves time.

This step eliminates requests that are not worth doing without a lot of effort on your part. Note if you eliminate on average 1 out of 40 projects at this step you’ve save time. However be honesty in your evaluation because you don’t want to exclude a high value project because you grossly over estimate it.

Step 2. What’s it’s worth?

What you do.

This is a step that is usually only done on large project but, as I will show it needs to be done on all projects. Given your design from the previous step what is the value of the new system directly in dollars and indirectly in intangible benefits. Take the time to ask some simple question like: What are the benefits of new system? Will be increase sales or cut cost? Will it save time, make someone job easier or less boring? Will it help to get a product to market sooner or just get information quicker? Having taken a look at its look at its benefits then ask the question does the benefits **far** out weigh the cost. I’ve emphasized far because we are in a resource scarce

business and if this project does not “obviously” create massive benefits then you need to move to something that will. Our goal here is to maximize your value and one obvious why is to make sure that your involve in only the best highest value projects.

How long should you spend.

This step should take from a few hours to a couple of weeks for the range of projects we are considering here. Sometimes this step can be a project in itself. But is important to note that you don’t have to know down to the last dollar the value of the new system because we are looking to do only high value system with an obvious benefit. If your value estimates are off a little, we still should be able to identify high value projects.

How this step saves time.

Even though this is probably an addition step for most people. It is one of the most important. It will save time by eliminate low or no value projects very early. The last thing you want to do is build a system that has no value. At this point you’ve spent 5% of the project time if you eliminate 1 out of 20 your time ahead.

Step 3. What are the risks?

What you do.

At this point in the project you need to determine the chances of success. There are three factors that determine the probability that a project will succeed they are:

1. The complexity of the technology involved
2. The expertise of the people doing the work
3. The support the project has.

Give each of these categories a number from 0 10 based on the following:

Category 1 → 0 is leading edge technology that
Has never been tried
10 for twenty year old proven
Technology.

Category 2 → 0 for novices
10 for experts.

Category 3 → 0 for no support
10 for the a managers support for
the small project or the president
of the company for the large
project.

Add the three numbers together.

The projects with a these total have the following chance of success.

25 to 30	Excellent
18.to 24	Good
13 to 17	Fair
below 12	Poor.

Given these ratings wait the potential return versus the risks. High risk should have a corresponding high return.

If not you may need to beef up the category that is low either a less technologically difficult design, more expertise or more support.

How long should you spend.

The amount of time for this step should not be very much if you use the simplified approach I’ve outlined here.

How this step saves time.

The function of this step is twofold. One this gives the client at an earlier point in the project a sense of the risks involved. And secondly to eliminate or correct at the projects start possible factors that can cause failure later on which is major time and value waster.

Step 4. Do A Very Detail Design and Schedule.

What you do.

This is the step where real work begins. Sometime this step is skip and the programming step begins using the preliminary step design to go by. My experience is that this is were a lot project get into big trouble. First of all because it is a rough estimate and the design is not tied down clearly, the estimate can be off by a factor of 10 which usually makes for a very unhappy client. I’m recommend that you do the detail design at this point in the project because it’s the best time to do it. At some point your going to have to nail down every detail to deliver the system why not do it at a point that it does you the most good and at a point where changes have the least impact.

One question is how detailed a design do you do. I suggest that you have enough detail that a programmer can do their job without needing any more information. Doing the design at a programmer level or data flow is too inefficient. It’s like programming the system twice. Normally I build process flow diagrams and have all the supporting information such as data dictionaries, file definitions and report layouts that a programmer would needs to their job. I also create an estimate and schedule of how long each step should take.

How long should you spend

This is a major step for the project and should take from 10% to 20% of the total project time.

How this step saves time.

This is a very important step and should not be skip. It saves time by determining the exact cost and resources required build the system. If the ballpark estimate is really off then the client needs to know now so they can cancel the project if its not worth it or resoucrs are not available. Most projects get into trouble because of bad estimates. The better the design the better the estimate the better the project will go. Also I’ve found that the clearer

the design the less time you'll spend on programming step which the largest step of the project.

Step 5. Build the System.

What you do.

For this step I like a "code to the max" approach. If you've done the previous steps properly this step which requires the most work should go smoothly and rapidly. If it's not going smoothly then you need to revisit the design, expertise or your support.

How long should you spend

This step is where most of the work is done and should take from 40% to 60% of the project. Too little time here and value of the project will suffer. Too much time and you're not at max effectiveness.

How this step saves time.

This is the step that everyone must do. Again if you've done the work leading to this step you should truly be able to build a system rapidly that you know is high value and not a waste of time because it's never used.

Step 6. Have weekly progress reports.

What you do.

Based on the progress to make weekly reports to all the people that involve in the project. Include in the report should be a weekly summary of progress which includes an estimate of the number of days the project is either ahead or behind based on the estimate in step 4. I also include the following files:

1. Open issues file.
2. Unexpected problems file.
3. A changed to original design.
4. A suggested enhancement file.

How long should you spend

This again is a very important step that is usually skipped or done occasionally. You should spend from 4 hours/week to 2 days a week.

How this step saves time.

The value of this step is that there are no surprises at the end of the project. You're building credibility during the whole project and if the project is over budget you've got justification for it. The time save is that the project is not canned at the last minute because it overruns and you don't get enough time to fix it.

Step 7. Do systematic testing.

What you do.

This is the first of three testing steps. I've divided testing into three steps to emphasize its importance. In this step the programmer checks for obvious problems. I call system bugs those that affect the whole system. There usually are very obvious and can be found by asking: Do the results make sense? Or by doing a ballpark estimate of what results should be.

How long should you spend

This step should spend from 2 of days to a couple of weeks.

How this step saves time.

Since this is a rather quick first crack at testing you should get a sense of how well the program is structured and programmed.

If you find a lot of problems you can easily go back and do some restructuring. The intent here is to get the system bug free before it goes into production. Fixing bugs after the system is in production takes four to ten times more time than at this point.

Step 8. Do transaction level testing.

What you do.

For the second level of testing you manually trace several transactions or the lowest level of data through the system to make sure every step is correct. This step will check every detail of the system.

How long should you spend

This step should spend from 1 week to a man month.

How this step saves time.

No system is complete until it is 100% bug free if you face that at this point and do the testing now you save a ton of time later. You going to spend at least 15% of the time fixing bugs you need to do it where you the most effective and now is the time.

Step 9. Do End-User testing.

What you do.

The last level of testing has two purposes, I've found that users are the most effective testers. They are the fastest bug finders available. For this step have an end-user take a look at the results and see if they can detect any problems.

How long should you spend

This step should spend from 2 days to a week.

How this step saves time.

This step saves time because it starts the transition process and end-users are the best tester going. They can find bugs like no other and in less time than no other can. But be careful not to over use them.

Step 10. Document it quickly.

What you do.

This step is only important from a standpoint of supporting and enhancing the system once it done. I recommend the following be done. All programs should be self-documented and be done as you build the program. Included also should be the final process flow diagrams. User documentation should be minimal if any at all.

How long should you spend

This step should spend from 2 or three days to a week.

How this step saves time.

Everyone wants lots of documentation but no one uses it. So don't spend a lot of time doing it. You should provide enough documentation that a new knowledgeable programmer can make changes to the system. If the system requires a lot of user documentation's the system is in trouble.

Step 11. Have a party.

What you do.

This is a very consequential step that I learned from a Six 6 company. They always have a party at the end of every project for a lot of substantial reasons. One is it fortifies the project as being a big success. It also gives everyone something to look forward to. To have a party you have to have an end date and get user acceptance. The end date is important because it forces the project to a conclusion. Some projects go on forever, wasting a lot of time. And some projects never really get accepted by the end-users creating lots of problems. I say have a party.

How long should you spend

One Lunch hour.

How this step saves time.

It is very important from a success and value standpoint that the system be used by the end-user. This step encourages the end-users to use the system and also not to let it drag on.

Step 12. What is the final value of the system?

What you do.

After you've had the party and the systems been in use for a while determine the true value of the system.

How long should you spend

A few days.

How this step saves time .

To be honest this is a step that is rarely done but I believe is the most meaningful.

This is the step where you learn what you did right and what you did wrong and what it was worth. So that the next time you can do a better faster job with more value.

Tom Mannigel is President of Insyst, Inc. a Houston based SAS Institute Quality Partner and has been a SAS system developer since 1981. His company has built over hundred different systems with a 97% success rate. He can be contact at tmannigel@aol.com.